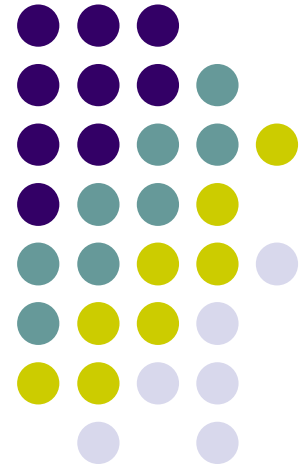


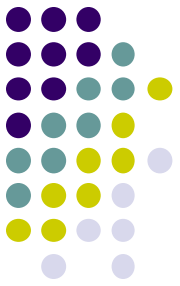
WIX1002

Fundamentals of Programming

Chapter 2

Java Fundamental





Contents

- Variable
- Operator
- Type Casting
- String
- Console Input
- Console Output
- Comment
- Random Number



Variable

- A variable is a storage location in the memory that has a **type, name and contents**.
- A variable must be **declared**, specifying the variable's **name** and the **type** of information that will be held in it.
- **Identifier** is the name of the variable. In Java, identifier may contains letters, digits (0 – 9), and the underscore (_) character.
- **However, identifier cannot begin with a digit, cannot contain spaces and cannot be reserved words.**
- A meaningful identifier will made the program easy to understand.
- By convention, variable names should start with a lowercase letter



Variable

- Java has basic types for characters, integers and floating points number. These basic types are known as **primitive types**.
- Java Unicode character chart
 - <http://www.ssec.wisc.edu/~tomw/java/unicode.html>

Java Primitive Data Types				
Type	Values	Default	Size	Range
byte	signed integers	0	8 bits	-128 to 127
short	signed integers	0	16 bits	-32768 to 32767
int	signed integers	0	32 bits	-2147483648 to 2147483647
long	signed integers	0	64 bits	-9223372036854775808 to 9223372036854775807
float	IEEE 754 floating point	0.0	32 bits	+/-1.4E-45 to +/-3.4028235E+38, +/-infinity, +/-0, NaN
double	IEEE 754 floating point	0.0	64 bits	+/-4.9E-324 to +/-1.7976931348623157E+308, +/-infinity, +/-0, NaN
char	Unicode character	\u0000	16 bits	\u0000 to \uFFFF
boolean	true, false	false	1 bit used in 32 bit integer	NA



Variable

- **declaration**

`int number;`

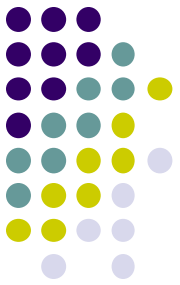
`double first, second;`

`boolean status;`

- **initialization**

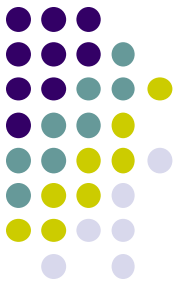
`double speed = 2.6;`

`char letter = 'A'; //use single quote for character`



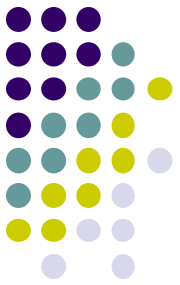
Variable

- A **constant** is an identifier that is similar to a variable except that it holds one value for its entire existence
- The compiler will issue an error if you try to change a constant variable
- Constant is declared using the **final** keyword.
 - `final int MIN=0;`
 - `final int MAX=100;`



Operator

- Operators are special symbols that perform specific operations on one or more **operands**.
- An operand is the part of a computer instruction which specifies what data is to be manipulated or operated on.
- **Assignment Operator (=)**
 - It is used to change the value of variable after the variable has been declared.
temperature = 36.9;
grade = 'c';
status = true;



Operator

- **Arithmetic Operator**

- It is used to compute numeric results
- + (addition) - (subtraction) * (multiplication) / (divison)
% (modulo or remainder)

temperature = a + 36.9;

total = monthlyPayment * 12;

answer = 20 % 6; // answer is 2

Operator



- Java follows **precedence rules** that determine how the operators are executed in order

Operator Precedence

Operators	Precedence
postfix	<code>expr++ expr--</code>
unary	<code>++expr --expr +expr -expr ~ !</code>
multiplicative	<code>* / %</code>
additive	<code>+ -</code>
shift	<code><< >> >>></code>
relational	<code>< > <= >= instanceof</code>
equality	<code>= !=</code>
bitwise AND	<code>&</code>
bitwise exclusive OR	<code>^</code>
bitwise inclusive OR	<code> </code>
logical AND	<code>&&</code>
logical OR	<code> </code>
ternary	<code>? :</code>
assignment	<code>= += -= *= /= %= &= ^= = <<= >>= >>>=</code>



Operator

- **Parentheses Operator ()**
 - Controls the order in which the operators execute in the expression. Parentheses override the normal precedence order
- **Postfix Operator**
 - `number++;`
 - **Use the current value of number.** Then increment by 1 for the next statement.
 - `number--;`
 - **Use the current value of number.** Then decrement by 1 for the next statement



Operator

- **Unary Operator**

- ++number;
 - **Increment number by 1** and then use the value
- --number;
 - **Decrement number by 1** and then use the value.

- **Other assignment operator**

- number += 5; // -=, *=, /=, %=
- number = number + 5;



Type Casting

- Sometimes it is convenient to convert data from one type to another. For example, we may want to treat an integer as a double value during a computation
- Type casting takes a value of one type and produces a value of another type.

```
int a=8, b=3;
```

```
double answer;
```

```
answer = a / b;
```

```
// Output is 2
```

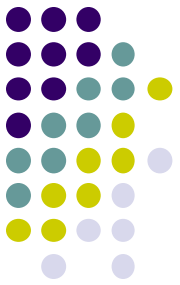
```
answer = a / (double) b;
```

```
// Output is 2.6666
```



String

- A string is a sequence of characters that is treated as a single value.
- String class is used to store and process **strings of characters**.
- Declaration and initialization
 - String fullname;
 - String topic = "Object-oriented Programming";
- Concatenation
 - + operator is used to connecting two strings.
 - String firstName, lastName, fullname;
 - fullname = firstname + lastname;

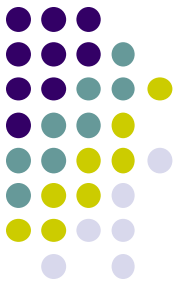


Console Input

- **Scanner Class**

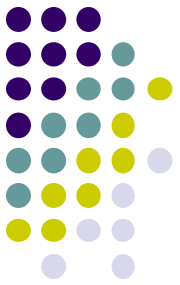
- A simple text scanner which can be used to **get input for primitive types and strings.**
- To load Scanner Class into java program
import java.util.Scanner;

```
Scanner keyboard = new Scanner(System.in);  
int num;  
System.out.println("Please enter a number");  
num = keyboard.nextInt();  
System.out.println("The number is " + num);
```



Console Input

- `.nextInt()`
 - Reads one int from the keyboard
- `.nextLong()`
 - Reads one long int from the keyboard
- `.nextDouble()`
 - Reads one double from the keyboard
- `.next()`
 - Reads **one word** into String class from the keyboard
- `.nextLine()`
 - Reads an entire line into String class from the keyboard



Console Output

- `System.out` is known as the standard output object
- **`System.out.println`** display a line of text in the command window. When it completes its tasks it automatically positions the output cursor to the beginning of the next line.
- **`System.out.print`** display a line of text in the command window. However, it does not position the output cursor at the beginning of the next line in command window.
- **`System.out.printf`** display output in a **specific format**



Console Output

Code	Output	Examples
d	decimal integer	%d %6d
f	fixed-point floating point	%f %6.2f
e	E-notation floating point	%e %4.2e
g	general floating point	%g %4.2g
s	string	%s %20s
c	character	%c %5c
-	left alignment	%-5d

- `System.out.printf("%6.2f", 22/7.0)`
 - Display 6 spaces with 2 decimal places as in `__3.14`
- `System.out.printf("PI%-6.2fValue", 22/7.0)`
 - Display `PI3.14__Value`



Console Output

- You can use some of the special characters in the standard output object
- Special Character
 - `\n` is the **newline character**. It cause the screen's output cursor to move to the beginning of the next line
 - `\t` is the **horizontal tab** character
 - `\\` display a backslash character
 - `\"` display a double quote character



Comment

- Comment is used to help other programmers to understand the program
- Comments **are not executed** when the application start.
- There are three types of comment in Java
- Single line comment
 - `//` A Single Line Comment
- Multiple lines comment

`/*`

A multiple
line comments

`*/`



Comment

- Documentation comment describe the functionalities of each Java class and methods.

```
/**
```

This method display a line of text on the screen

```
*/
```

- How to write a standard Java Documentation Comment
 - <https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html>
- To run javadoc on a package
 - **javadoc -d documentationDirectory PackageName**



Random Number

- Random number is very useful in game and simulation
- The Random class of Java implements a random number generator
 - **import java.util.Random;**

```
Random g = new Random();
```

```
int num;
```

```
num = g.nextInt();    //any random value of integer
```

```
int MAX = 100;
```

```
num = g.nextInt(MAX); // random value from 0 to 99
```

